CMPT 135-D100 Midterm Exam 1
Spring 2023

This is a **50 minute closed book exam**: notes, books, computers, calculators, electronic devices, etc. are **not** permitted. Do not speak to any other students during their exam or look at their work. Please remain seated and **raise your hand** if you have a question.

Pointers and Memory Management

a) (5 marks) Write a function called `range(int n)` that returns a pointer to a new `vector<int>` (allocated on the free store) that contains the numbers 0, 1, 2, … , `n`-1.

For example, it should work like this:

```
vector<int>* v = range(5);
for (int n : *v) {
    cout << n << " ";
}
// prints: 0 1 2 3 4

delete v;
```

b) (5 marks) Write a function called `deallocate_both(vector<int>* a, vector<int>* b)` that correctly de-allocates the `vector<int>`s that `a` and `b` point to. Assume that `a` and `b` point to vectors created by one or more calls to `range` (from question a). It's also possible that one, or both, of `a` and `b` could be `nullptr`.

`deallocate_both` should correctly de-allocate the vectors `a` and `b` point to and not any memory errors (or other kinds of errors).

For example:

```
vector<int>* v1 = range(5);
vector<int>* v2 = range(10);
deallocate_both(v1, v2);
// v1 and v2 have been de-allocated

vector<int>* v3 = range(5);
vector<int>* v4 = nullptr;
deallocate_both(v3, v4);
// v3 has been de-allocated

vector<int>* v5 = range(5);
deallocate_both(v5, v5);
// v5 has been de-allocated
```

Write your answer here:

```
void deallocate_both(vector<int>* a, vector<int>* b)
{
```

Object-oriented Programming and Inheritance

a) (5 marks) Add the following to the `Movie` class below:

1. A **copy constructor** that uses an **initialization list** to make a new `Movie` object that is a copy of another `Movie` object.
2. A **destructor** that prints “done!”.
3. A **setter** that lets the user change the name of a `Movie`.
4. A **getter** that returns the year of a `Movie`.

```
class Movie {
    string title;
    int year;

public:
    Movie(const string& t, int y) {
       title = t;
       year = y;
    }

    // ... your code goes here ...
```

b) (5 marks) Write a class call `int_list` that has all the features of a `vector<int>`, and also has a non-const method called `first()` that returns an `int` pointer to the first element of the vector. If the `int_list` is empty, `first()` should return `nullptr`.

For example:

```
int_list lst;
lst.push_back(5);
lst.push_back(10);
cout << lst[0];       // prints 5
cout << lst[1];       // prints 10
int* p = lst.first();
cout << *p;           // prints 5
```

Multiple Choice

For each of the following questions, fill in **the one best answer** on the answer sheet.

Every correct answer is worth 1 mark. Incorrect answers, unanswered questions, questions with more than one answer, or questions with illegible answers, are worth 0.

1) Consider these two statements:

   i) A { corresponds to a “push” onto the call stack
   ii) A } corresponds to a “pop” on the call stack

   Which one of these statements most accurately describes the truth values of i) and ii)?

   A. i) and ii) are both true
   B. i) and ii) are both false
   C. i) is false and ii) is true
   D. i) is true and ii) is false

2) Where are **local** variables stored?

   A. only the call stack
   B. only the free store
   C. sometimes the call stack, sometimes the free store
   D. only static memory

3) Consider these two statements:

   i) Blackbox tests can be created *without* seeing the implementation of a function.
   ii) Whitebox tests can be created *without* seeing the implementation of a function.

   Which one of these statements most accurately describes the truth values of i) and ii)?

   A. i) and ii) are both true
   B. i) and ii) are both false
   C. i) is false and ii) is true
   D. i) is true and ii) is false

4) Consider these two statements:

i) `assert(2 == 2)` crashes the program when it runs.
ii) `assert(2 != 2)` crashes the program when it runs.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is false and ii) is true
D. i) is true and ii) is false

5) Consider these two statements:

i) Only constructors can have initialization lists.
ii) A class can have multiple constructors and multiple destructors.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is false and ii) is true
D. i) is true and ii) is false

6) Consider these two statements:

i) Immutable objects have no setters.
ii) A non-const getter will always cause a compiler error.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is true and ii) is false
D. i) is false and ii) is true

7) Consider these two statements:

i) For a child class to be able to re-implement a method it inherits from a parent class, the method must be declared **virtual in the parent class.**
ii) For a child class to be able to re-implement a method it inherits from a parent class, the method must be declared **virtual in the child class**.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is false and ii) is true
D. i) is true and ii) is false

8) Consider these two statements:

i) For a child class to be able to re-implement a method it inherits from a parent class, the method must be declared **abstract in the parent class.**
ii) For a child class to be able to re-implement a method it inherits from a parent class, the method must be declared **abstract in the child class**.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is false and ii) is true
D. i) is true and ii) is false

9) Consider this statement:

```
int n = 0;  // line 1
```

Consider these two statements:

i) Line 1 is an example of a **definition**.
ii) Line 1 is an example of a **declaration**.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is false and ii) is true
D. i) is true and ii) is false

10) Consider these two statements:

i) C++ functions must be **declared** exactly once.
ii) C++ functions can be **defined** more than once.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is false and ii) is true
D. i) is true and ii) is false

11) These two functions can appear in the same namespace:

```
namespace ui {
      void code(int n) {
         // …
      }

      void code(int n, int t) {
         // …
      }

      // …
}
```

A. True
B. False

12) Consider these two statements:

i) A single `using` statement can give access to **a single function** in a namespace.
ii) A single `using` statement can give access to **every** function in a namespace.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is false and ii) is true
D. i) is true and ii) is false

13) Consider this C++ function:

```
void f() {
    int* p = new int[100];

    throw std::runtime_error("error!"); // line 1

    delete[] p;                         // line 2
}
```

Consider these two statements:

i) `f` has a memory leak.
ii) If you move line 1 so it comes *after* line 2, then `f` has a memory leak.

Which one of these statements most accurately describes the truth values of i) and ii)?

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is false and ii) is true
D. i) is true and ii) is false

14) What is the correct way for a try/catch block to catch *all* exceptions that `g()` might throw?

A.
```
try {
   g();
} catch () {
   // …
}
```
B.
```
try {
   g();
} else {
   // …
}
```
C.
```
try {
   g();
} catch (...) {
   // …
}
```
D.
```
try {
   g();
} default {
   // …
}
```
E. None of the above.

15) A pointer of type `double*` can point to itself.

A. True
B. False